JFW
AF

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of	
Inventors: Steven ROTH, et al.	: Confirmation No. 4656
	:
U.S. Patent Application No. 09/875,289	: Group Art Unit: 2116
	:
Filed: June 7, 2001	: Examiner: E. CHANG
	:
For: DYNAMIC KERNEL TUNABLES	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Attn: BOARD OF PATENT APPEALS AND INTERFERENCES

Sir:

Further to the Notice of Appeal filed June 22, 2005, and responsive to the Notice of Non-Compliant Appeal Brief mailed November 1, 2005, in connection with the above-identified application on appeal, herewith is Appellants' Amended Brief on Appeal. The statutory fee of $500 was paid on August 22, 2005.

To the extent necessary, Appellants hereby request any required extension of time not otherwise requested and hereby authorize the Commissioner to charge any required fees not otherwise provided for, including application processing, extra claims, and extension fees, to Deposit Account No. 08-2025.

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Cases

I. Real Party in Interest

The real party in interest is Hewlett-Packard Company.

II. Related Appeals and Interferences

There are no related appeals and/or interferences.

III. Status of Claims

No claims are allowed.

Claims 1-30 are rejected under 35 USC 102(b) as being anticipated by the preface of "AIX for Breakfast" by Houtz.

IV. Status of Amendments

There are no unentered amendments.

V. Summary of Claimed Subject Matter

The claimed subject matter concerns modifying operating system behavior by changing tunable parameters for an operating system, and more specifically, dynamic kernel tunable parameters. Kernel tunable parameters are described in the instant specification as "variables . . . that control the operation of the kernel, that are defined in the master files and set in the system files, and that are compiled into the kernel" (Instant specification at page 12, lines 23-25). More particularly, a "dynamic tunable parameter is a tunable parameter whose value can be changed by a system administrator without requiring a reboot of the system to effect the change." (Instant specification at page 13, lines 3-5). The claimed subject matter claims modifying kernel behavior by changing the data on which the operating system instructions operate: the values of certain variables called "tunables." In other words, the claimed subject matter discusses modifying the behavior of the kernel without modifying the actual software of the kernel.

The present claimed subject matter discloses, in one embodiment, a method of updating tunables used in a kernel which remain persistent across computer reboots.

A system file is updated including tunables having a tunable setting with a new tunable value in response to a single administrator request. The adminstrator "can tune the OS kernel controlling the operation of system 210 directly without a reboot" and "can update a kernel registry service 220 and the system files 225 using the application called kmtune 234, which uses the UNIX command line and provides a common user interface to the tunable parameters and the persistence mechanism." (Instant specification at page 14, lines 1-5 and FIG. 2). Also, the administrator "can use the system administration manager (SAM) 230, which is a graphical user interface. SAM 230 translates the administrator's actions into invocations of the kmtune

application." (Instant specification at page 14, lines 5-8). The kmtune application is able to directly change values in the kernel registry service and the system files by using a system call interface 232 which includes three calls: settune() 242, tuneinfo () 244 and gettune () 246. (Instant specification at page 14, lines 8-12).

Updating means for updating a system file includes a set of executable instructions comprising the system administration manager (SAM) graphical user interface, the kmtune operating system command, and an individually accessible portion of the dynamic kernel tunables program, i.e., the system call settune.

A persistent storage mechanism is simultaneously updated including tunables having a tunable setting with the new tunable value in response to the single administrator request. "Two mechanisms are used to ensure that tunable value changes remain persistent: the system files 225 and the Kernel Registry Service (KRS) 220." (Instant specification at page 15, lines 22-23). Any persistent parameter storage device can be used, e.g., a disk file, disk, EEPROM, or other persistent storage mechanism can be used. All tunable value changes made using either SAM, kmtune, or settune remain persistent when the kernel reboots. (Instant specification at page 16, lines 8-11).

Updating means for simultaneously updating a persistent storage mechanism includes the SAM graphical user interface, the kmtune operating system command, and the settune call. "Once a change has been validated and executed, the tunable information in the Kernel Registry Service 220 will be updated to reflect the change." (Instant specification at page 19, lines 8-10).

A tunable value is changed in the kernel with the new tunable value and the computer continues to run with the updated tunable value. "All tunable value changes made . . . will

remain persistent when the kernel reboots." (Instant specification at page 16, lines 8-9). "[K]ernel sub-system tunables can be changed through the handler function 260 without a reboot." (Instant specification at page 14, lines 17-18). In particular with respect to the kmtune application, when kmtune is invoked with the -u flag, in addition to changing the system file, kmtune also changes the value of the tunable used by the currently running kernel without requiring a kernel rebuild or reboot. (Instant specification at page 15, lines 14-16).

Changing means for changing a tunable value include the SAM graphical user interface, the kmtune operating system command, and the settune call. "SAM 230 translates the administrator's actions into invocations of the kmtune application." (Instant specification at page 14, lines 7-8). Kmtune changes "the value of the tunable being used by the currently running kernel." (Instant specification at page 15, lines 15-16. The settune API 242 can be used to change tunable parameters remaining effective until the next time the kernel is rebuilt. (Instant specification at page 15, lines 17-19).

The present claimed subject matter discloses, in another embodiment, a method of rebuilding a kernel.

Tunable settings are retrieved which are stored in a system file. "The system files 225 contain the tunable settings that are used when a kernel gets built." (Instant specification at page 16, lines 21-22). Retrieving means for retrieving tunable settings are known operating system mechanisms, e.g., referenced by file system tunable handler 266.

The kernel is rebuilt using the retrieved tunable settings and a persistent storage mechanism is updated using the retrieved tunable settings. "When a kernel gets rebuilt, the

tunable data in the KRS 220 is erased and replaced with the tunable data in the system files 225." (Instant specification at page 17, lines 1-2).

Advantageously, high availability of the operating system is achieved and the system is usable with updated tunables without having to reboot the system. (Instant specification at page 25, lines 13-14). Correspondingly, behavioral modifications to the kernel may be made without software changes, i.e., by adjusting tunable parameters, and in a way that results in no service interruption at all.

In fact, the present specification differentiates the present claimed subject matter from dynamically loadable kernel modules by stating that even with DLKM "the kernel must be rebuilt and rebooted in order for the tunable value changes to take effect." (Instant specification at page 2, line 29 - page 3, line 6). See also, page 15, lines 12-13 of the instant specification "changing a DLKM tunable the DLKM must be unloaded, recompiled and reloaded to incorporate the new tunable value."

Additionally, as described in the instant specification, prior approaches to dynamic tunables used one mechanism to change a tunable value while the system is running and a separate and different mechanism to change the value in a permanent fashion to last across reboots. (Instant specification at page 3, lines 14-17).

VI. Grounds of Rejection to be Reviewed on Appeal

A. Houtz fails to anticipate claims 1-30

VII. Argument

A. Houtz fails to anticipate claims 1-30

Houtz describes an approach to modifying kernel behavior different from the present claimed subject matter and as such fails to anticipate the claimed subject matter.

Houtz describes modifying kernel behavior by way of adding or replacing modules, i.e., services, with respect to the kernel. (Houtz at paragraph 6). In fact, the Houtz system appears to operate similarly to the dynamically loadable Kernel modules differentiated in the instant specification at page 2, line 29 – page 3, line 6. Further, Houtz describes a tool, i.e., the system management interface tool (SMIT), used to accomplish the interaction of a user with the operating system to modify the kernel by binding "new service(s) into the 'kernel' of the operating system." (Houtz at paragraph 4, lines 2-3). Nowhere does Houtz disclose kernel tunables or the modification thereof.

A rejection based on 35 U.S.C. §102 requires every element of the claim to be included in the reference, either directly or inherently. The Houtz reference ("AIX for Breakfast" by Phillip Houtz) fails to anticipate claim 1 as Houtz fails to include all elements of claim 1. There are at least three reasons Houtz fails to anticipate claim 1.

1. Houtz fails to describe kernel tunables

As described above, nowhere in Houtz is there any mention of kernel tunables. Specifically, Houtz fails to disclose updating tunables used in a kernel as claimed in claim 1, i.e., "updating a system file including tunables," "updating a persistent storage mechanism including tunables," and "changing a tunable value in the kernel."

The Examiner asserts that Houtz discloses system parameters affecting kernel behavior at paragraph 2, lines 10-13; however, this is incorrect. The word "parameters" as used at the Examiner-cited location of Houtz refers to parameters of a SMIT task being invoked and not to parameters affecting kernel behavior. That is, the default parameters

mentioned relate to parameters of the SMIT task being requested and not to modifying kernel tunable parameters. SMIT task parameters are unrelated to kernel tunables.

Further, Houtz fails to support the proposition that the SMIT task parameters are "edited and recompiled into a new kernel that is subsequently booted and used. [paragraph 4]" (Advisory Action mailed June 8, 2005 at page 2, paragraph 1). Additionally, paragraph 4 referenced by the Examiner is unrelated to the SMIT task-based system described by Houtz and inapplicable to support the Examiner's assertion. Paragraph 4 describes recompilation and rebooting of a kernel in order to bind new services into the kernel.

At most, Houtz describes dynamically binding services to an operating system kernel without requiring a reboot. "As a result, AIX 3 can unbind a service and/or bind a new or newly configured version of a service." Houtz at paragraph 6. However, services added dynamically to a kernel are not the same as modifying tunables in the kernel. Houtz describes editing files and recompiling UNIX in the context of modifying the kernel and not in the context of modifying behavior of the kernel by changing the data variables controlling operation of the kernel.

The Examiner asserts that "Houtz teaches that the process of dynamically adding a newly configured version of a service [paragraph 6, lines 8-12] comprises updating tunable parameters used by the kernel [paragraph 10, lines 6-10]." (Advisory Action mailed June 8, 2005 at page 2, paragraph 2). The Examiner is incorrect as Houtz describes binding a newly configured version of a service and not changing a tunable parameter. For example, paragraph 10 of Houtz discusses binding a device driver to the kernel, i.e., changing the kernel software, and not updating a tunable parameter. Changing the kernel software requires that the system stop using the software for a brief time during the change, and in many cases, the entire system needs to be rebooted. The resulting stoppage interrupts the provision of services performed by the system. In contrast, the present claimed subject matter modifies kernel behavior without software changes requiring stoppage and results in no service provision interruption.

As described above, Houtz fails to disclose kernel tunables as claimed in the claimed subject matter. Based thereon, the claimed subject matter is patentable over Houtz and the rejection should be reversed.

2. Houtz fails to inherently describe kernel tunables

Further, the Examiner has failed to meet the burden required to establish a case of inherency. If the Examiner is stating that the quoted portion of Houtz inherently provides kernel tunables, the Examiner has not met the burden of establishing a prima facie case of inherency.

The fact that a certain result or characteristic may occur or be present in the prior art is not sufficient to establish the inherency of that result or characteristic. In re Rijckaert, 9 F.3d 1531, 1532, 28 U.S.P.Q.2d 1955, 1956 (Fed. Cir. 1993); In re Oelrich, 666 F.2d 578, 581-82, 212 U.S.P.Q. 323, 326 (C.C.P.A. 1981). To establish inherency, extrinsic evidence must make clear that the missing descriptive matter is necessarily present in the thing described in the reference and that it would be so recognized by persons of ordinary skill in the art. Inherency may not be established by possibilities or probabilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient. In re Roberston, 169 F.3d 743, 745, 49 U.S.P.Q.2d 1949, 1950-51 (Fed. Cir. 1999). In relying upon a theory of inherency, the Examiner must provide a basis in fact or technical reasoning to reasonably support the determination that the allegedly inherent characteristic necessarily flows from the teachings of the prior art. Ex parte Levy, 17 U.S.P.Q.2d 1461, 1464 (B.P.A.I. 1990).

Since the Examiner has not provided a rationale or evidence to show that the quoted portion of Houtz inherently provides kernel tunables, the rejection of claim 1 based on Houtz is incorrect and should be reversed.

3. Houtz fails to describe a single administrator request causing the update of a system file

Nowhere does Houtz disclose a single administrator request causing the update of a system file including tunables. As described supra, Houtz at paragraphs 2-4 describes the addition of services to the kernel but not the update of system tunables, i.e., "'bind' the new service(s) into the 'kernel' of the operating system." Houtz at paragraph 4. The Examiner has failed to identify any location in Houtz describing a single administrator request causing the update of a system file including tunables.

Based on the foregoing, the claimed subject matter is patentable over Houtz and the rejection should be reversed.

4. Houtz fails to disclose simultaneous update of a persistent storage mechanism in response to a single administrator request

Houtz fails to disclose simultaneously updating a persistent storage mechanism including tunables in response to the above-described single administrator request. Houtz at paragraph 10 describes the addition of a service to the kernel and not the update of a system file with a new tunable value. No system file tunable value is updated in Houtz.

Further, Houtz fails to disclose tunable values updated in persistent storage and in a system file in response to an administrator request. Houtz only describes the editing of files and linkages to add services to an operating system.

Based on the foregoing, the claimed subject matter is patentable over Houtz and the rejection should be reversed.

For each of the above reasons, claim 1 is patentable over Houtz and the rejection should be reversed.

Claims 2-11 depend from claim 1, include further important limitations, and are patentable over Houtz for at least the reasons advanced above with respect to claim 1. The rejection of claims 2-11 should be reversed.

With respect to claim 12, Houtz fails to disclose kernel tunable settings for at least reasons similar to those advanced above with respect to claim 1, and further, Houtz fails to disclose updating a persistent storage mechanism using tunable settings as claimed in claim 12. As described above, paragraphs 2-4 of Houtz disclose binding services and not manipulating kernel tunables. The statement that "Houtz teaches that the tunable value in the kernel is changed on a per-service basis without recompiling the entire kernel. Thus, each tunable can be made changed dynamically without needing any centralized interface changes" is incorrect as Houtz only discloses kernel service bindings. For at least this reason, claim 12 is patentable over Houtz and the rejection should be reversed.

Claim 13 depends from claim 12, includes further important limitations, and is patentable over Houtz for at least the reasons advanced above with respect to claim 12. The rejection of claim 13 should be reversed.

With respect to claim 14, Houtz fails to disclose kernel tunable settings for at least reasons similar to those advanced above with respect to claim 1, and further, Houtz fails to disclose a handler function interface. Houtz at paragraph 6 discloses service binding operations and not kernel tunables. The Examiner was requested to identify where in paragraph 6 Houtz discloses the asserted handler function interface. The Examiner has failed to identify any supporting teaching in Houtz and as no such interface exists, and for the reasons advanced above, the rejection of claim 14 should be reversed.

Claim 15 depends from claim 14, includes further important limitations, and is patentable over Houtz for at least the reasons advanced above with respect to claim 14. The rejection of claim 15 should be reversed.

Claims 16, 18, and 20 are patentable over Houtz for reasons similar to those advanced above with respect to claim 1 and the rejection should be reversed.

Claims 17, 19, and 21 are patentable over Houtz for reasons similar to those advanced above with respect to claim 12 and the rejection should be reversed.

Claims 22-30 depend, either directly or indirectly, from claim 1 and include further important limitations. Claims 22-30 are patentable over Houtz for at least the reasons advanced above with respect to claim 1 and the rejection should be reversed.

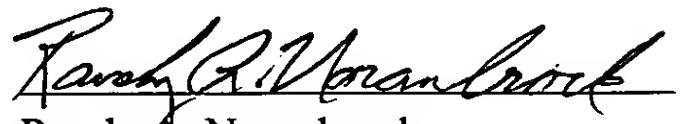
VIII. Conclusion

For the extensive reasons advanced above, the present claimed subject matter of claims 1-30 are patentable over Houtz and the rejection of claims 1-30 should be reversed.

Reversal of the rejection is in order.

Respectfully submitted,

Steven ROTH, et al.

By: 
Randy A. Noranbrock
Reg. No. 42,940

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, CO 80527-2400
Telephone: 703-684-1111
Facsimile: 970-898-0640

KMB:RAN/iyr

IX. Claims Appendix

1. A method of updating tunables used in a kernel which remain persistent across computer reboots, comprising:

updating a system file including tunables each having a tunable setting with a new tunable value in response to a single administrator request;

simultaneously updating a persistent storage mechanism including tunables each having a tunable setting with the new tunable value in response to the single administrator request;

changing a tunable value in the kernel with the new tunable value without needing any centralized interface changes and continuing to run the computer with the updated tunable value.

2. The method of claim 1, wherein said updating steps are performed using a system administrator's management application.

3. The method of claim 1, wherein updating steps are performed using a UNIX command line.

4. The method of claim 1, wherein the computer includes a UNIX operating system.

5. The method of claim 1, wherein the system file includes a file for a core kernel and another for each separately loadable kernel module.

6. The method of claim 1, further comprising compiling the kernel and incorporating the new tunable value into the kernel.

7. The method of claim 1, wherein the persistent storage mechanism is a Kernel Registry Service.

8. The method of claim 1, comprising retrieving a current value of a tunable setting.

9. The method of claim 1, comprising retrieving detailed information about one or more tunable settings.

10. The method of claim 1, comprising registering handler functions for a tunable handler for a particular tunable.

11. The method of claim 1, comprising interfacing with the kernel using a handler function.

12. A method of rebuilding a kernel, comprising:

retrieving tunable settings stored in a system file;

rebuilding the kernel using the retrieved tunable settings; and

updating a persistent storage mechanism without needing any centralized interface changes using the retrieved tunable settings.

13. The method of claim 12, further comprising:

erasing data stored in a persistent storage mechanism;

replacing the erased data with the retrieved tunable settings.

14. A dynamic kernel tunable framework for changing tunables in a kernel without rebooting, comprising:

a graphical user interface for displaying and changing graphical values and settings of dynamic tunables;

a system call interface for interfacing the user interface with a system file, a persistent storage mechanism and the kernel;

a handler function interface interfaced to the system call interface and the kernel including information about each dynamic tunable without needing any centralized interface changes.

15. The framework of claim 14, in which tunable changes are made immediately without rebooting and are also kept persistent across reboots, both actions taken as the result of a single administrator request.

16. A computer architecture, comprising:

updating means for updating a system file including tunables each having tunable setting with a new tunable value in response to a single administrator request;

updating means for simultaneously updating a persistent storage mechanism including tunables each having tunable setting with the new tunable value in response to the single administrator request; and

changing means for changing a tunable value in the kernel with the new tunable value without needing any centralized interface changes and continuing to run the computer with the updated tunable value.

17. A computer architecture, comprising:

retrieving means for retrieving tunable settings stored in a system file;

rebuilding means for rebuilding the kernel using the retrieved tunable settings; and

updating means for updating a persistent storage mechanism without needing any centralized interface changes using the retrieved tunable settings.

18. An article, comprising:

at least one sequence of machine executable instructions;

a medium bearing the executable instructions in machine readable form, wherein execution of the instructions by one or more processors causes the one or more processors to:

retrieve a system file including tunables each having tunable setting with a new tunable value in response to a single administrator request;

simultaneously retrieve a persistent storage mechanism including tunables each having tunable setting with the new tunable value in response to the single administrator request; and

change a tunable value in the kernel with the new tunable value without needing any centralized interface changes and continuing to run the computer with the retrieved tunable value.

19. An article, comprising:

at least one sequence of machine executable instructions;

a medium bearing the executable instructions in machine readable form, wherein execution of the instructions by one or more processors causes the one or more processors to:

retrieve tunable settings stored in a system file;

rebuild the kernel using the retrieved tunable settings; and

update a persistent storage mechanism using the retrieved tunable settings without needing any centralized interface changes.

20. A computer system, comprising:

a processor; and

a memory coupled to said processor, the memory having stored therein sequences of instructions, which, when executed by said processor, causes said processor to perform the steps of:

update a system file including tunables each having tunable setting with a new tunable value in response to a single administrator request;

simultaneously update a persistent storage mechanism including tunables each having tunable setting with the new tunable value in response to the single administrator request; and

change a tunable value in the kernel with the new tunable value without needing any centralized interface changes and continuing to run the computer with the updated tunable value.

21. A computer system, comprising:

a processor; and

a memory coupled to said processor, the memory having stored therein sequences of instructions, which, when executed by said processor, causes said processor to perform the steps of:

update tunable settings stored in a system file;

rebuild the kernel using the retrieved tunable settings without needing any centralized interface changes; and

update a persistent storage mechanism using the retrieved tunable settings.

22. The method of claim 1, wherein the tunables can be made independently of each other.

23. The method of claim 1, wherein new tunables can be created without needing any centralized interface changes.

24. The method of claim 1, wherein a data structure is maintained that includes information about every tunable parameter.

25. The method of claim 24, comprising compiling the data structure into the kernel to initialize the data structure in a kernel registry.

26. The method of claim 1, wherein each tunable has an associated handler function.

27. The method of claim 1, comprising updating a tunable using SAM.

28. The method of claim 1, comprising updating a tunable using kmtune.

29. The method of claim 1, comprising updating a tunable using a kernel system call (API).

30. The method of claim 1, comprising using one of a SAM, kmtune and a kernel system call to update a tunable.

X. Evidence Appendix

None.

XI. Related Proceedings Appendix

None.